文章编号: 1001-0920 (2011) 12-1818-06

带分批优化的多级批处理过程自组织调度方法

梁 涛[1,2], 李歧强[1]

(1. 山东大学 控制科学与工程学院，济南 250061；2. 山东电力工程咨询院有限公司，济南 250013)

摘 要: 针对一类带批次划分的多级批处理过程优化调度问题, 提出一种自下而上的自组织调度方法. 首先, 通过构造与批处理生产过程中的订单、批次和设备相对应的自组织个体, 建立自组织调度模型框架; 然后, 分析多级批处理调度问题的最优性质, 提出分批优化规则和自组织选择策略, 并在此基础上给出自组织优化调度算法; 最后, 通过调度实例求解结果表明, 所提方法能在短时间内获得问题的最优解或近优解, 进而验证了该方法的有效性和优越性.

关键词: 分批优化；批处理；调度；自组织

中图分类号: TP391.9 **文献标识码:** A

Self-organizing approach to multistage batch scheduling with batching optimization

LIANG Tao[1,2], *LI Qi-qiang*[1]

(1. School of Control Science and Engineering，Shandong University，Ji'nan 250061，China；2. Shandong Electric Power Engineering Consulting Institute Corporation Ltd，Ji'nan 250013，China. Correspondent：LIANG Tao，E-mail：liangtao@mail.sdu.edu.cn)

Abstract: A bottom-up self-organizing scheduling approach is presented to optimize a kind of scheduling problems with batching optimization in multistage batch processes. Firstly, a self-organizing scheduling model framework is built up by constructing kinds of self-organizing units associated with real-world orders, batches and equipment units in the batch processes. The optimal properties of multistage batch scheduling problems are analyzed. Then, batching optimization rules and self-organizing selection strategies are introduced in detail. Based on the strategies, a self-organizing optimal scheduling algorithm is proposed for the given model. Finally, several examples are given and the computational results show that the presented approach can obtain optimal solutions or near-optimal solutions in a short time, which verifies the effectiveness and the superiority of the proposed approach.

Key words: batching optimization；batch process；scheduling；self-organization

1 引 言

多级批处理过程作为流程工业中一种典型的生产形式, 在制药[1]、食品[2]和石油化工[3-4]等行业中得到了广泛的应用. 它固有的灵活性决定了可通过合理的调度达到增产降耗和节能减排等目的. 近年来, 国内外学者针对多级批处理过程的调度方法作了大量的工作, 并获得了很大的进展[5-7]. 然而, 绝大部分研究都是假定调度问题中的批次数目是已知的, 而且是固定的. 在实际生产中, 由于不同容量并行设备的存在, 在给定产品需求的情况下, 批次数目和批量大小在调度过程中都是可变的, 如果按照固定批量对需求进行批次划分, 往往会导致调度结果的失优. 针对这类问题, Prasad 等人[8-9]提出了一种能同时优化批次选择、分配与排序的混合整数线性规划 (MILP) 模型, 在一定程度上可有效解决此类问题. 但是, 随着问题复杂性的增加和问题规模的增大, 该方法往往面临着复杂性难于描述和模型难于求解等问题.

自组织调度方法是将实际生产过程看成一种自组织过程, 在一定的自组织规则下, 通过自组织个体 (SOU) 的动态交互协同, 自下而上生成优化调度结果的一种方法[10]. 它通过构造与实体对应的自组织个体和确定合适的自组织规则, 从而达到自组织生成调度方案的目的. 自组织调度方法较他组织调度方法具有更强的驾驭复杂性的能力和更好的自动趋优能

收稿日期: 2010-08-05；**修回日期:** 2010-09-29.

基金项目: 国家 863 计划项目 (2007AA04Z157).

作者简介: 梁涛(1983–), 男, 博士, 从事复杂系统建模与优化的研究；李歧强(1964–), 男, 教授, 博士生导师, 从事复杂系统建模与优化算法、节能优化技术等研究.

力. 本文在研究了分批优化规则的基础上, 提出一种适合解决带分批优化的多级批处理过程调度问题的自组织优化方法, 并通过应用实例表明了所给出方法的有效性和优越性。

2　问题描述

本文研究一类多级多产品的批处理过程调度问题, 给出一组处在 K 个生产级上的 M 个批处理设备 $\{u_j|j = 1,2,\cdots,M\}$ 和 N 个产品订单 $\{o_i|i = 1,2,\cdots,N\}$, 如图 1 所示. 每个生产级 k 上可有多个并行设备 $\{u_j|j \in J_k, k \leqslant K\}$, 各批处理设备的加工容量上下限 $b_j^{\max}$ 和 $b_j^{\min}$ 给定. 各产品订单需要划分为一个或多个批次进行生产, 已知各订单 o_i 的订货量 q_i, 投放时间 r_i 和交货时间 d_i, 以及其在各个批处理设备 u_j 上的批次处理时间 τ_{ij} 和允许的加工路径. 调度问题的目标就是在满足生产约束和订单需求的条件下, 找出一个最优的批次划分和排产方案使得最大完工时间最短.

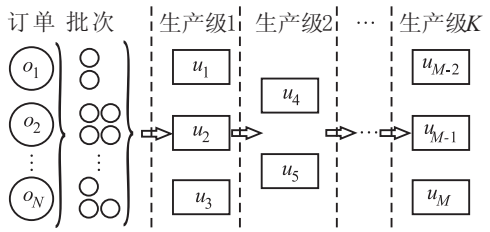

图 1　带批次划分的多订单多级批处理过程

3　自组织调度方法

3.1　自组织模型框架

多级多产品的批处理生产过程一般包含 3 类实体: 订单、批次和批处理设备. 通过构造与批处理生产过程中的各个实体相对应的自组织个体 (SOU), 可将实际生产过程看成一个自组织过程, 如图 2 所示.

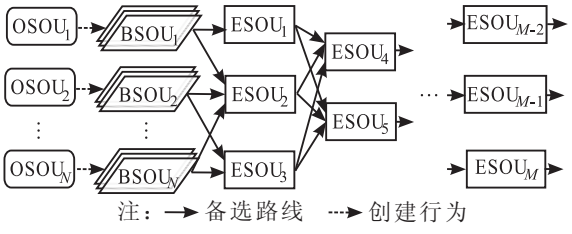

图 2　带分批优化的多级批处理过程自组织调度模型框架

该自组织模型中包含订单 SOU(OSOU), 批次 SOU (BSOU)和设备 SOU(ESOU) 共 3 类 SOU. 这 3 类 SOU 作为自主个体, 具有自身的属性和行为. 其中, 每个 OSOU, BSOU 和 ESOU 分别对应实际生产中的一个订单、批次和设备, 其属性包含了对应实体的生产信息. OSOU 可根据一定的分批规则创建相应的 BSOU; BSOU 可对 ESOU 进行选择, 并到达相应的 ESOU 等待处理; 每个 ESOU 具有一个等待批次队列, 等待其处理的 BSOU 位于该队列中, ESOU 可从等待队列中选择合适的 BSOU 进行处理. 这 3 类 SOU 的属性和行为可以方便地应用面向对象的方法来创建. 该模型中的各类 SOU 也可根据需要方便地添加、移除或更新信息, 因此该模型具有很强的建模灵活性和可扩展性. 在该自组织模型框架下, 自组织调度的成败主要取决于自组织规则的合理性.

3.2　分批优化规则

在实际批处理生产过程中, 一个订单的满足往往需要多个批次的生产, 而每个生产批次的批量都不能超出其所在生产设备容量的上下限. 合理的分批方案是保证优化排产、实现总体调度目标的保证.

针对第 2 节中描述的多级批处理问题, 设按生产顺序从各个生产级的并行批处理设备中每级选出一个设备组成一条加工路径, 记为 p_l, 其设备集合记为 JP_l, l 为加工路径编号; 设备 u_j 的加工批量记为 b_j, 加工路径 p_l 上的加工批量记为 $\hat{b}_l$; 对订单 o_i 允许的加工路径集合记为 P_i, 所有允许的加工路径的集合记为 P, $P = P_1 \bigcup P_2 \bigcup \cdots \bigcup P_N$, 所有被禁止的加工路径的集合记为 $\bar{P}$; 该调度问题的可行解集记为 S, 最优解集记为 $S^*(S^* \subseteq S)$, 则可得以下定理:

定理 1　加工路径 p_l 上所有生产批次的批量上限为 $\hat{b}_l^{\max} = \min\limits_{j\in \mathrm{JP}_l}(b_j^{\max})$, 批量下限为 $\hat{b}_l^{\min} = \max\limits_{j\in \mathrm{JP}_l}(b_j^{\min})$. 若 $\hat{b}_l^{\max} < \hat{b}_l^{\min}$ 成立, 则有 $p_l \in \bar{P}$ 成立.

证明　实际生产中, 对于 $\forall u_j, j = 1,2,\cdots,M$, 有 $b_j^{\min} \leqslant b_j \leqslant b_j^{\max}$ 成立. 显然, 对于 $\forall u_j, j \in \mathrm{JP}_l$, 也有 $b_j^{\min} \leqslant b_j \leqslant b_j^{\max}$ 成立. 因为任意生产批次都有 $\hat{b}_l = b_j$ 对于 $\forall j \in \mathrm{JP}_l$ 成立, 则有 $b_j^{\min} \leqslant \hat{b}_l \leqslant b_j^{\max}$ 对于 $\forall j \in \mathrm{JP}_l$ 成立, 所以 $\max\limits_{j\in \mathrm{JP}_l}(b_j^{\min}) \leqslant \hat{b}_l \leqslant \min\limits_{j\in \mathrm{JP}_l}(b_j^{\max})$ 成立. 由此可得, $\hat{b}_l^{\max} = \min\limits_{j\in \mathrm{JP}_l}(b_j^{\max})$, $\hat{b}_l^{\min} = \max\limits_{j\in \mathrm{JP}_l}(b_j^{\min})$; 若 $\hat{b}_l^{\max} < \hat{b}_l^{\min}$, 则 $\hat{b}_l$ 无法同时满足 $\hat{b}_l \leqslant \hat{b}_l^{\max}$ 和 $\hat{b}_l \geqslant \hat{b}_l^{\min}$, 所以, $\hat{b}_l$ 不存在可行解, 显然有 $p_l \in \bar{P}$ 成立. □

定理 2　若 S^* 为非空集合, 则 $\exists s \in S^*$ 使 s 中 $\forall p_l \in P$ 上批次的批量均为 p_l 上的批量上限 $\hat{b}_l^{\max}$.

证明　设 $s' \in S^*$, 其目标值为 κ. 将 s' 中所有加工路径 $p_l \in P$ 上批次的批量均增大到批量上限 $\hat{b}_l^{\max}$, 得到调度方案 s. 因为各批次在批处理设备上的处理时间 τ_{ij} 给定且为常数, 所以, s 与 s' 两方案中各批次在各设备上处理的起止时间均相同. 显然, $s \in S$, 且其目标值 $\kappa^* = \kappa$, 可得 $s \in S^*$, 由此定理可证. □

定理 3　若 $\exists s_1, s_2 \in S$, s_1 和 s_2 的目标值分别为 κ_1 和 κ_2, s_1 是在其批次划分条件下的最优调度方案, s_1 和 s_2 中订单 o_i 的批次个数分别为 Z 和 $Z+1$, 除

o_i 以外各订单的批次划分情况和除 o_i 的多余批次外的各批次在设备上的分配情况 s_2 与 s_1 相同, 则有 $\kappa_1 \leqslant \kappa_2$ 成立.

证明 s_1 中订单 o_i 的各批次的批量值记为 b_1^i, $b_2^i, \cdots, b_Z^i$, s_2 中订单 o_i 的前 Z 个批次与 s_1 中订单 o_i 的 Z 个批次在设备上的分配情况相同, 多余批次记为第 $Z+1$ 个批次. 删除 s_2 中订单 o_i 的第 $Z+1$ 个批次, 并将订单 o_i 剩余 Z 个批次的批量对应 s_1 中订单 o_i 的批次分别调整为 $b_1^i, b_2^i, \cdots, b_Z^i$, 得到一个新的调度方案 s_2', 其目标值记为 κ_2'. 由此可得: $\kappa_2' \leqslant \kappa_2$ 且 s_2' 中订单 o_i 各批次的总批量 $\sum\limits_{z=1}^{Z} b_z^i \geqslant q_i$, $s_2' \in S$ 且 s_2' 的批次划分情况与 s_1 相同. 因为 s_1 是在其批次划分条件下的最优调度方案, 则有 $\kappa_1 \leqslant \kappa_2'$ 成立. 由 $\kappa_1 \leqslant \kappa_2' \leqslant \kappa_2$ 可得 $\kappa_1 \leqslant \kappa_2$, 即 s_2 的目标值不优于 s_1. □

由定理 1∼定理 3 可以得出以下结论: 为了生成最优调度方案, 在选定加工路径的条件下, 批次的批量可以等于该加工路径上的批量上限值, 而且在特定加工路径下, 应以最少的批次数来满足订单需求. 由此构造分批优化算法如下:

算法 1 分批优化策略如下:

Step 1: 为每个 ESOU 定义一个预定队列, 初始化预定队列为空, 队列完成时间 $t_j^* = 0$.

Step 2: 随机选择一个 $q_i > 0$ 的 OSOU. 被选中的 OSOU 根据完成时间倒数概率原则 (基于概率

$$p_j^* = [1/(t_j^* + \tau_{ij})] \Big/ \sum_{j' \in J_k \bigcap JA_i} [1/(t_{j'}^* + \tau_{ij'})],$$

从设备集合 $J_k \bigcap JA_i$ 中选择设备 u_j 所对应的 ESOU, 其中 JA_i 为允许处理订单 o_i 的设备集合) 逐级选出一个 ESOU 构成加工路径 p_l, 并计算其批量上限值 $\hat{b}_l^{\max}$.

Step 3: 被选中的 OSOU 创建一个批量为 $\hat{b}_l^{\max}$ 的 BSOU, 并更新订单需求, 即 $q_i = q_i - \hat{b}_l^{\max}$; 若 $q_i < 0$, 则 $q_i = 0$.

Step 4: 更新选定加工路径 p_l 中 ESOU 的预定队列完成时间, 即 $t_j^* = t_j^* + \tau_{ij}, j \in JP_l$.

Step 5: 若存在 $q_i > 0 (i = 1, 2, \cdots, N)$ 的 OSOU, 则转到 Step2; 否则, 结束.

3.3 自组织选择策略

分批优化算法中, OSOU 根据 ESOU 的预定队列状态为其创建的每一个 BSOU 预选了一条加工路径. 但在自组织调度过程中, 由于订单投放时间的不同和伴随着批次的处理完成, ESOU 的等待批次队列与预定队列存在着很大差异, 预定的加工路径往往变得不够优化. 因此, 各 ESOU 和 BSOU 需在自组织调度过程中, 根据实际情况实时地进行选择处理或被处理, 这将更加有利于最优解的涌现.

一方面, BSOU 从允许的候选设备中选择哪一个 ESOU, 主要与设备的状态和设备对该批次的处理能力有关. 设备 u_j 在 t 时刻的状态定义为

$$\mu_j(t) = A_j(t) \Big/ \Big(1 + \sum_{x=1}^{Q_j(t)} \tau_{i_x j}\Big). \tag{1}$$

其中: $A_j(t) \in \{0, 1\}$ 为 t 时刻设备 u_j 的可用性, 可用为 1, 不可用为 0; $Q_j(t) \in$ 自然数集, 为 t 时刻设备 u_j 的等待批次队列中批次的个数; i_x 为设备 u_j 的等待批次队列中第 x 个批次的订单编号; 分母中 1 的作用是将 $\mu_j(t)$ 的值域限定在 [0, 1]. 设备状态 $\mu_j(t)$ 是表征设备可用性与忙闲状态的参量, 式 (1) 将两者统一在该参量中. $\mu_j(t)$ 的值越大, 说明完成设备 u_j 等待批次队列中的批次任务所需的时间越少, 则设备 u_j 被选中的几率会越大; 反之, 设备 u_j 被选中的几率会越小. 设备对批次的处理能力主要与设备处理该批次所需的处理时间 τ_{ij} 有关. 为了实现 BSOU 对 ESOU 的合理选择, 其对 ESOU 的偏好概率可表示为

$$p_j^E = \frac{\delta_j(\mu_j)^\alpha (1/\tau_{ij})^\beta}{\sum\limits_{j' \in U_{ik}} \delta_{j'}(\mu_{j'})^\alpha (1/\tau_{ij'})^\beta}. \tag{2}$$

其中: p_j^E 为候选 ESOU 中设备 u_j 对应的 ESOU 被选中的概率; $\delta_j \in \{0, 1\}$, 也可记为 $\delta_j(b)$, 为进行选择的 BSOU 批量大小 b 的函数, 表示设备 u_j 是否能够处理该批量的 BSOU, 若 $b_j^{\min} \leqslant b \leqslant b_j^{\max}$ 成立为 1, 否则为 0; U_{ik} 为 BSOU 所属订单 o_i 在第 k 级被允许的候选 ESOU 的编号集合; α, β 为两个参数, 取值范围均为 [0, 2], 分别反映了设备的当前状态和设备对该批次的处理能力在设备被选中过程中的相对重要性, 它们的取值可根据经验或者通过仿真实验确定. 若实际问题中设备的忙闲状态对问题目标的影响大于设备对该类批次的处理能力, 则取 $\alpha > \beta$; 反之, 则取 $\alpha < \beta$. 仿真实验表明, 针对一般实际问题, 在无特殊要求或尚无经验可循的情况下, 可取 $\alpha = \beta = 1$.

另一方面, ESOU 从其等待批次队列中选择哪一个 BSOU 先处理, 主要与 BSOU 对应批次所属订单的投放时间 r_i 和批次的处理时间裕量有关. 设备 u_j 的等待批次队列中订单 o_j 的批次在时刻 t 的处理时间裕量定义为

$$tr_{ij}(t) = d_i - t - \tau_{ij} - \sum_{k=k_j+1}^{K} \Big(\frac{1}{W_{ik}} \sum_{y=1}^{W_{ik}} \tau_{ij_y}\Big). \tag{3}$$

其中: k_j 为设备 u_j 所处的生产级, W_{ik} 为订单 o_i 在第 k 级上被允许的候选设备集合 U_{ik} 中的设备数量, j_y 为候选设备集合 U_{ik} 中第 y 个设备的编号. 批次的处理时间裕量表征了批次被加工的紧迫性, 其值越小, 该批次越应该被优先加工. 由此, ESOU 对它的等待

批次队列中BSOU的偏好概率可表示为

$$p_x^B=\frac{\sigma_{xj}/tr_{i_xj}}{\dfrac{Q_j(t)}{\displaystyle\sum_{x'=1}(\sigma_{x'j}/tr_{i_{x'}j})}}. \tag{4}$$

其中: p_x^B 为设备 u_j 等待批次队列中第 x 个BSOU被选中的概率; $\sigma_{xj}\in\{0,1\}$ 为设备 u_j 等待批次队列中第 x 个BSOU的投放标志, 若 $t\geqslant r_{i_x}$ 成立取1; 否则取0. ESOU只被允许选择已被投放的BSOU, 即满足 $t\geqslant r_{i_x}$ 的BSOU.

3.4 自组织调度优化算法

基于分批优化规则和自组织选择策略, 下面给出改进的自组织调度优化算法.

算法 2　Step 1: 初始化自组织模型中的各SOU属性和状态, 将当前最优目标值置为 $+\infty$, 给出自组织循环次数值 $\bar{N}$ 和各个经验参数取值.

Step 2: 根据算法1对所有订单 o_i 进行批次划分, 得到需要处理的BSOU.

Step 3: 各ESOU的响应时间置为一个极大数 H ($H>\max\limits_i(d_i)$), Step 2中得到的所有BSOU以随机顺序按其偏好概率选择生产级1上的ESOU, 并到达ESOU的等待批次队列, 被选中的ESOU更新状态, 响应时间置为 $\min(\hat{t}_{\rm sel},r_{i^*})$. 其中: $\hat{t}_{\rm sel}$ 为被选中ESOU当前的响应时间, r_{i^*} 为当前执行选择的BSOU对应订单的投放时间.

Step 4: ESOU按照响应时间顺序执行自组织行为, 若存在当前处理批次, 则完成当前处理的批次, 然后按照其偏好概率从等待批次队列中选择合适的BSOU进行处理. 如果选中, 则将其响应时间设置为被选中BSOU的完成时间; 否则, 置为 H (对于生产级1上的ESOU, 若等待批次队列不为空, 则置为等待批次的最小投放时间), 更新设备状态. 被完成的BSOU按其偏好概率选择下一生产级上的ESOU, 并到达下一级ESOU的等待批次队列, 被选中的下一级ESOU更新状态, 若其响应时间等于 H, 则置为当前处理时间; 若无当前处理批次, 则ESOU直接执行选择BSOU和更新响应时间和状态. ESOU的执行顺序随响应时间的更新而更新, 直到所有BSOU都已完成第 K 级的处理任务或者ESOU的响应时间都为 H.

Step 5: 如果Step 4中获得的结果满足订单交货期要求, 并且所得目标值小于当前最优目标值, 则保存该结果作为当前最优调度结果; 否则, 保持当前最优调度结果不变.

Step 6: $\bar{N}=\bar{N}-1$, 如果 $\bar{N}>0$, 则重新初始化各SOU属性和状态, 转到Step 2.

Step 7: 输出最优调度结果, 算法结束.

为了分析自组织调度优化算法的收敛性, 设最优解集 S^* 中所有调度方案对应的分批方案构成最优分批方案集 Γ, 则最优解对应着最优分批条件下的最优资源分配和任务排序方案. 算法1产生的分批方案属于集合 Γ 的概率记为 p^Γ, 其产生的所有分批方案组成的集合记为 F. 下面给出算法收敛定理:

定理 4　基于算法1与式(2)和(4)所表征的自组织选择策略的自组织调度优化算法以概率1收敛于全局最优.

证明　考虑到算法1中各加工路径 $p_l\in P$ 被选中的概率均大于0, 根据定理2和定理3容易得出 $p^\Gamma>0$. 设 $f\in\Gamma\bigcap F$, 其在集合 $\Gamma\bigcap F$ 所有分批方案中产生几率的比重记为 $p^f(p^f>0)$. 在分批方案 f 下, 算法2执行一次Step 3和Step 4, 获得某一最优资源分配和任务排序方案的概率可表示为

$$\boldsymbol{p}_1^f=\prod_k\prod_i\prod_z\frac{\delta_{\hat{j}}(\mu_{\hat{j}})^\alpha(1/\tau_{i\hat{j}})^\beta}{\displaystyle\sum_{j'\in U_{ik}}\delta_{j'}(\mu_{j'})^\alpha(1/\tau_{ij'})^\beta}\cdot$$
$$\prod_j\prod_{\hat{x}}\frac{\sigma_{\hat{x}j}/tr_{i_{\hat{x}}j}}{\dfrac{Q_j(t)}{\displaystyle\sum_{x'=1}(\sigma_{x'j}/tr_{i_{x'}j})}}. \tag{5}$$

其中: $\hat{j}$ 为 $\hat{j}_{zik}$ 的缩写, $\hat{x}$ 为 $\hat{x}_j$ 的缩写, 分别为该最优资源分配和任务排序方案下生产级 k 上订单 o_i 的第 z 个批次分配的设备编号和设备 u_j 上各批次的次序编号. 由最优资源分配和任务排序方案下 $\delta_{\hat{j}}=1$, $\mu_{\hat{j}}>0$ 和 $\sigma_{\hat{x}j}=1$, 可得 $\boldsymbol{p}_1^f>0$. 每一个最优分批方案下存在一个或多个最优资源分配和任务排序方案, 因此在分批方案 f 下, 算法2执行一次Step 3和Step 4, 能够获得最优资源分配和任务排序方案的概率 $\boldsymbol{p}^f\geqslant\boldsymbol{p}_1^f>0$. 由算法2中ESOU的响应时间连续更新策略容易得出, 算法2在分批方案 f 下生成的最优资源分配和任务排序下的调度方案即为问题的最优解. 因此, 算法2自组织循环一次获得最优解的概率可表示为 $\boldsymbol{p}_1=p^\Gamma\sum\limits_{f\in\Gamma\bigcap F}p^f\boldsymbol{p}^f$. 显然 $0<\boldsymbol{p}_1\leqslant1$, 则算法2自组织循环 $\bar{N}$ 次获得最优解的概率可表示为 $\boldsymbol{p}_{\bar{N}}=1-(1-\boldsymbol{p}_1)^{\bar{N}}$, 由 $0<\boldsymbol{p}_1\leqslant1$ 可知, $\lim\limits_{\bar{N}\to+\infty}\boldsymbol{p}_{\bar{N}}=1$, 因此, 在 $\bar{N}\to+\infty$ 时, 算法2以概率1收敛于全局最优. □

由以上证明过程可知, 自组织选择策略中经验参数的取值一定程度上影响着算法的收敛速度, 如果经验参数设置不合理, 则该算法可能出现找不到满意解的情况. 在此种情况下, 应该适当调整经验参数取值或者适当增大 $\bar{N}$ 值后重新计算.

4 实例求解与分析

带分批优化的多级批处理过程自组织模型框架

和自组织调度算法采用C++进行编程实现, 在CPU为2.66 GHz, 内存为1 GB的PC机上进行求解. 为了验证自组织调度方法的有效性, 本文对5个不同规模的实例进行建模和求解. 实例1[9]中装置包含了2个生产级, 每级有3个批处理设备, 装置结构与设备参数如表1所示. 表1中, 禁止路径包括(u_1, u_6)和(u_2, u_5). 有8个产品订单在此装置上生产, 其处理时间、订货量、投放时间和交货时间分别如表2和表3所示.

表 1 装置结构与设备参数表

生产级	设备	$b_j^{\min}$/kg	$b_j^{\max}$/kg
1	u_1	10	25
1	u_2	10	25
1	u_3	15	30
2	u_4	10	25
2	u_5	15	30
2	u_6	10	25

表 2 订单处理时间

订单	设备/h					
	u_1	u_2	u_3	u_4	u_5	u_6
1	-	6.5	9.4	5.2	6.7	5.0
2, 3	7.3	6.5	9.4	5.2	6.7	5.0
4, 5, 6	6.5	7.5	8.5	4.5	7.0	4.8
7, 8	6.8	7.5	9.0	4.8	6.5	5.0

表 3 订单参数

参数	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
q_i/kg	30	30	30	20	20	20	20	20
r_i/h	0	0	0	0	0	0	0	0
d_i/h	40	50	50	40	30	40	60	60

实例2∼实例5为4个随机实例, 其订单的订货量、投放时间和交货时间为随机产生, 生成策略如下:

For exam=2 to 5

For $i = 1$ to 8

$q_i = 10 + 50\times$exam$\times$random(0, 1);

If random(0, 1)>1/3

$r_i = 0$;

Else

$r_i = 30\times$random(0, 1);

$d_i = r_i + q_i\times$random(0.3, 1);

End

End.

其他参数与实例1相同.

应用本文提出的自组织调度方法, 各经验参数设置为$\alpha = \beta = 1$, 自组织循环次数$\bar{N}$针对5个实例的规模分别设置为: 5 000, 5 000, 10 000, 20 000和20 000, 优化计算结果如表4所示.

表 4 优化调度结果比较表

实例	自组织调度方法			基于MILP的方法[9]	
	N	目标值/$	时间/s	目标值/$	时间/s
1	5 000	30.8	2.79	30.8	5.1
2	5 000	44.0	3.34	44.0	382.4
3	10 000	56.8	8.16	56.8	2 419.6
4	20 000	98.6	20.79	103.5	10 000*
5	20 000	102.3	26.64	124.6	10 000*

注: * 经过相应的时间结束计算, 将最佳可行解作为优化结果.

表4中同时给出了文献[9]提出的基于MILP模型的调度方法对5个实例在同一配置PC机上进行优化求解的结果, 求解器选用LINDO API 5.0. 从表4中可以看出, 对于规模较小的实例1∼实例3, 两种方法都可以找到问题的最优解, 而本文方法所用时间远远小于基于MILP模型的方法. 随着问题规模的增大, 最优解越来越难被找到. 对于较大规模的实例4和实例5, 自组织调度方法与基于MILP模型的调度方法相比, 在更短的时间内找到了更优的可行解. 这些主要归功于合理的自组织规则和自组织的涌现作用. 一方面, 自组织求解过程中, 各种生产约束被作为自组织规则而得到遵守, 从而缩小了需要搜索的解域范围, 绝大多数情况下, 算法可以直接在可行解域中搜索问题的最优解; 另一方面, 合理的自组织规则对最优解的获取具有很好的引导作用, 可以在短时间内获得问题的最优解或近优解. 这些特性决定了本文方法在求解速度上和规定时间内获得解的质量上都要优于以解域遍历为基础的传统算法. 实例的求解结果同时也验证了自组织规则的合理性. 另外, 从表4中可以看出, 自组织调度方法的求解时间随问题规模增长的速度远远小于文献[9]中给出的基于MILP模型的调度方法, 这主要是由于自组织调度算法中的自组织行为交互次数随着问题规模的增大呈近似线性增长, 使得算法的求解时间也随着问题规模的增大呈近似线性增长($\bar{N}$值固定), 这与MILP模型求解时间的指数级增长速度相比, 具有很大的优越性, 同时也使得本文方法在求解大规模问题时的优势愈加明显.

5 结 论

分批优化决策是多级批处理过程优化排产的一个重要方面, 将分批优化和批次调度分别孤立地进行决策容易导致排产结果偏离最优的排产方案. 本文针对多级批处理过程的特点, 提出了一种新型的适合对批次划分和调度集成优化的自组织调度方法. 本文方法基于给定的分批优化规则和自组织选择策略, 能够自下而上地涌现出问题的优化解. 实例计算结果验证了本文方法不但具有较快的求解速度, 而且具有较好的寻优性能. 此外, 本文方法建模灵活, 可扩展性强, 在解决多级批处理过程复杂调度问题领域具有较好的应用前景和推广价值.

参考文献(References)

[1] Stefansson H, Shah N. Multi-scale planning and scheduling in the pharmaceutical industry[C]. European Symposium on Computer-Aided Process Engineering-15. Barcelona: Elsevier, 2005: 1003-1008.

[2] Simpson R, Abakarova A. Optimal scheduling of canned food plants including simultaneous sterilization[J]. J of Food Engineering, 2009, 90(1): 53-59.

[3] Jia Z, Ierapetritou M. Efficient short-term scheduling of refinery operations based on a continuous time formulation[J]. Computers & Chemical Engineering, 2004, 28(6/7): 1001-1019.

[4] Qian Yu, Pan Ming, Huang Ya-cai. Modeling and optimization for scheduling of chemical batch processes[J]. Chinese J of Chemical Engineering, 2009, 17(1): 1-7.

[5] Floudas C A, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review[J]. Computers & Chemical Engineering, 2004, 28(11): 2109-2129.

[6] Méndez C A, Cerdá J, Grossmann I E, et al. State-of-the-art review of optimization methods for short-term scheduling of batch processes[J]. Computers & Chemical Engineering, 2006, 30(6/7): 913-946.

[7] Liu Y, Karimi I A. Scheduling multistage batch plants with parallel units and no interstage storage[J]. Computers & Chemical Engineering, 2008, 32(4/5): 671-693.

[8] Prasad P, Maravelias C T, Kelly J D. Optimization of aluminum smelter casthouse operations[J]. Industrial & Engineering Chemistry Research, 2006, 45(22): 7603-7617.

[9] Prasad P, Maravelias C T. Batch selection, assignment and sequencing in multi-stage multi-product processes[J]. Computers & Chemical Engineering, 2008, 32(6): 1106-1119.

[10] Liang Tao, Li Qi-qiang, Ding Ran. A self-organizing approach to reactive scheduling of multiproduct batch plants[C]. Proc of the 8th Int Conf on Computing, Communication and Control Technologies. Orlando: IIIS, 2010: 102-107.

(上接第1817页)

[9] Storn R, Price K. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[10] Nomana N, Iba H. Differential evolution for economic load dispatch problems[J]. Electric Power Systems Research, 2008, 78(8): 1322-1331.

[11] Coelho L, Souza R, Mariani V. Improved differential evolution approach based on cultural algorithm and diversity measure applied to solve economic load dispatch problems[J]. Mathematics and Computers in Simulation, 2009, 79(10): 3136-3147.

[12] Meng K, Wang G, Dong Z, Wong K. Quantum-inspired particle swarm optimization for valve-point economic load dispatch[J]. IEEE Trans on Power Systems, 2010, 25(1): 215-222.

[13] Vlachogiannis J, Lee K. Economic load dispatch — A comparative study on heuristic optimization techniques with an improved coordinated aggregation-based PSO[J]. IEEE Trans on Power Systems, 2009, 24(2): 991-1001.

[14] Pandi V, Panigrahi B. Bacterial foraging optimisation: Nelder — Mead hybrid algorithm for economic load dispatch[J]. IET Generation, Transmission & Distribution, 2008, 2(4): 556-565.

[15] Park J, Jeong Y, Shin J, et al. An improved particle swarm optimization for nonconvex economic dispatch problems[J]. IEEE Trans on Power Systems, 2010, 25(1): 156-166.

[16] Lee K, Sode-Yome A, Park J. Adaptive hopfield neural networks for economic load dispatch[J]. IEEE Trans on Power Systems, 1998, 13(2): 519-526.

[17] Park Y, Won J, Park J. A new approach to economic load dispatch based on improved evolutionary programming[J]. Int J of Engineering Intelligent Systems for Electrical Engineering and Communications, 1998, 6(2): 103-110.

[18] Chiang C. Improved genetic algorithm for power economic dispatch of units with valve-point effects and multiple fuels[J]. IEEE Trans on Power Systems, 2005, 20(4): 1690-1699.

[19] Selvakumar A, Thanushkodi K. A new particle swarm optimization solution to nonconvex economic dispatch problems[J]. IEEE Trans on Power Systems, 2007, 22(1): 42-51.

[20] Immanuel S, Thanushkodi K. Anti-predatory particle swarm optimization: Solution to nonconvex economic dispatch problems[J]. Electric Power Systems Research, 2008, 78(1): 2-10.

[21] Lu H, Sriyanyong P, Song Y, et al. Experimental study of a new hybrid PSO with mutation for economic dispatch with non-smooth cost function[J]. Int J of Electrical Power and Energy Systems, 2010, 32(9): 921-935.